Investigating the Free/Libre Open Source Software Commons in Commercial Organizations

Research-in-Progress

Juho Lindman
Hanken School of Economics
Arkadiankatu 22, 00101 Helsinki, Finland
juho.lindman@hanken.fi

Abstract

This paper contrasts the open-source promise and its actual implementations in organizations. It describes what kinds of discursive changes take place in commercial organizations when they co-opt open-source ways to produce software. This qualitative, interpretative case study in a comparative setting investigates whether such software production can be conceptualized as commons. Findings indicate that the meaning of the term "open source software" varies greatly in the studied cases and that the resulting way to produce software can hardly be characterized as commons or commons based.

Keywords: Open source software development, FLOSS, OSS 2.0, Inner source, Organizing vision

Introduction

Open source software is a term that was coined by practitioners (Perens, 1998; Raymond, 1999). Free/Libre Open Source Software (FLOSS) theorists, coming from a wide variety of scientific fields, have used FLOSS as an icon for a networked society (Castels, 2005), a communal resource (Von Krogh, 2002), an example of commons-based peer production (Benkler, 2007), an example of a cellular form of post-capitalist new order (Bauwens, 2005), or the new networked digital commons (Zizek, 2009).

In what follows, we continue this discussion by empirically showing how the term "open-source software" has been applied in commercial companies. This is important because the promise of FLOSS was to revolutionize software production and the industry. However welcome, this revolution has not (yet) materialized, although FLOSS products are everywhere. Instead, we are observing that FLOSS benefits (commercial) organizations by offering more open development processes and outcomes.

We focus on this tension between global expectations and local organizational practice. Our research question is as follows: What kinds of discursive changes took place when hierarchical and commercial organizations co-opted FLOSS? When investigating what happens to commons-based production, our findings showed local and sometimes dramatic transitions in the meaning of the term "open-source software." The re-emerging theme in our cases is the path these organizations take when they move away from commons-based, peer-to-peer production when implementing FLOSS.

Background

In this paper, engagement in organizational software production (with FLOSS) is conceptualized not as related to licenses (for more on licensing, see Välimäki, 2005) but rather as an organizational change related to the adoption of new, more open development tools and practices following the OSS 2.0 model (Fitzgerald, 2006). Practices include the use of e-mail (and archives) as the one primary communication tool, the availability of the code in a source-code repository, the web presence of a project (e.g., Sourceforge), the use of concurrent versioning system (CVS) or similar, and the use of issue trackers.

Many researchers have used FLOSS as an example of commons-based, peer-to-peer production (for example, Benkler, 2007). New technologies generate "new commons" (Hess & Ostrom, 2007). It is plausible to discuss how to limit deleterious use of natural resources. However, production of digital artifacts differs from the use of natural resources. Ostrom and Schlager (1996) suggested characterization of the different types of goods: (1) public goods (e.g., a sunset), (2) common-pool resources (e.g., a library), (3) club goods (e.g., a country club), and (4) private goods (e.g., a personal computer). Definitions of FLOSS in the public press have been directed at the twin audiences of commercial companies (Raymond, 2001; Scacchi, 2007) and FLOSS enthusiasts. It is, then, unsurprising that research literature found that "(FL)OSS is not a precise term" (Gacek & Arief, 2004, p. 35). Currently, it is even debatable which kinds of goods FLOSS artifacts belong to (especially as we omit discussion on licenses).

There are organizational opportunities that constrain FLOSS implementation. Due to these constraints, companies leverage FLOSS by implementing one of several "generic FLOSS business models" (Hecker, 1999; Osterwalder et al., 2005) or using different hybrid models (Sharma, 2002) or management strategies (Shaikt & Cornford, 2010). Organizations also engage in virtual collaboration with external FLOSS developers on projects that may reside outside their immediate control (Dahlander and Magnusson, 2008). Individual developers working for a company face a challenge of two competing logics: FLOSS and proprietary (Rolandsson et al., 2011).

When organizations adopt FLOSS, they rely on different carriers to develop local conceptual understanding (e.g., meetings, presentations, workshops, trainings, shops, and expositions). Organizing vision describes how organizational diffusion and legitimization takes place, focusing on the shared community discourse as the development engine of change. Past examples of these kinds of organizational visions include once buzz-wordish innovations such as CASE-tools, client-server, and intranet (Swanson & Ramiller, 1997). The concept of organizing vision stands for a "…a focal community idea for the application of information technology in organizations" (Swanson & Ramiller, 1997, p. 460).

This manuscript investigates how the shared understanding regarding FLOSS changes during the adoption process. The paper chooses to view FLOSS as an organizing vision. The focus is on understanding the changes of meaning of the term taking place locally. An organizing vision can be divided into three different aspects, forming the core of my following analyses: (1) interpretation, (2) legitimation, and (3) mobilization (Swanson and Ramiller, 1997). The interpretation represents efforts to create a "story" about the purpose of the term. Legitimation represents the underlying rationale for the organizing vision. Mobilization activates organizations to carry out the local implementation of the innovation in question (Swanson and Ramiller, 1997).

Methodology

We used a qualitative approach to adopt the protocol of interpretative case studies (Klein & Myers, 1999; Walsham, 1995). The actual field research was conducted as part of a European research project (ITEA) spanning 2005–2010. All the organizations were undergoing FLOSS-inspired changes in their software production and as participants of the project were interested in understanding the changes in their organization better.

The data emerged from semi-structured interviews complemented by industry-academia project meetings and joint work on project deliverables. We also used secondary material over the project duration. Information was gathered on the history of the cases, the organizational changes occurring over time, and current challenges. The topic of the thematic interviews was changes in software production and FLOSS practices, especially in relation to the current organizational opportunities and business models. The

interview protocol first addressed collecting basic information about the respondents and the history of the described cases. Business model change was the second topic under investigation. The questions concerned the resources, networks, offering, and revenue models of the organizations. Finally, time was left for discussion on emerging topics. The interviews resulted in a large corpus of data on the dynamics the organizations were going through.

Some of the respondents were interviewed several times over the duration of the project to see whether their description of the process changed over time. The interviews were about one hour long. In total, this resulted in 18 interviews provided by 12 people consisting of over 22 hours of recordings. Respondents were chosen from different organizational positions. For each case, one respondent was chosen from an internal software service unit, one from an internal business unit, and one from a developer/user perspective.

In the analyses, the focus was on how the respondents talked about the term FLOSS and about the three interlinked aspects of organizational vision: (1) interpretation, (2) legitimation, and (3) mobilization (Swanson & Ramiller, 1997). **Interpretation** was identified by analyzing how respondents described and defined "open-source software" in their organization. The interpretation aspect answers the question "What is FLOSS in your organization?" **Legitimation** was identified as the respondents' reasoning behind the good and bad sides (legitimate) of FLOSS for the organization. The legitimation aspect answers the question "What are good and bad sides of FLOSS in your organization?" **Mobilization** was identified when respondents discussed change in software production and its impacts on the different groups in the organization. The mobilization aspect thus answers the question "How did the software production change in organization and which groups benefited?" It was in some cases difficult to separate the three aspects when respondents talked about the subject, so in these cases, they were tabled in several places in the analyses that follows.

Findings

Case 1: Implementing Distributed Collaborative Software Development Practices Internally

The case unit's company is currently employing approximately 31,000 people and offers a portfolio of technical medical-equipment systems and related software. The customer base consists of medical professionals and patients. The company's products have a large installed base, and interoperability requirements make changes to existing software expensive. The case describes changes from a centralized software group into a more distributed development setting while promoting the visibility of the software assets.

Interpretation. In the earlier setting, software components were developed in a central software group and then integrated into products in different business units. Different business units had unaligned roadmaps, which made it difficult to forecast the necessary workload for the central software group and thus forced business units to wait for the excess capacity of the central group to provide them the — often hurried — software assets. "[…] *previously we were always waiting for platform groups to do things for us.* […] *If you wanted something done, you would queue up with all the other groups who would need something from the platform. And depending on how our demands were appreciated by the platform group, it would be in the next release or it would never get into release."* –Manager

To solve this issue, business units started to contribute to developing the software assets, building up their own expertise and then contributing to the shared portfolio. The change was described as moving to an inner source way of working. "*Internally we have a supply chain, where software components, in an Inner Source fashion, are being supplied to system groups*." –Manager

Legitimation. "[…]*say you have multiple groups with the past of developing the software on their own island. Now, we suddenly say that we have to work together […] It is an idea that needs some selling. It is not automatically done. It is not happening in one go. You can talk about it, we need to do this, we need to work for Inner Source.*" –Manager. FLOSS technology (tools and practices) was seen a way to make this change more legitimate to the managers and developers. "*In our [case], community is not Open Source but much more like inner source. I mean, what our model is used internally within [our company]*?" –Program Manager

Even from the beginning, the meaning of FLOSS changed to inner source. "*On a number of aspects we have quite a formal relationship. We have steering groups, we make agreements, we make project plans, we give commitments to project plans.*" –Program Manager

The application of the variant of the term open source was now one of wider visibility of source code between the central group and the business units and also a new division of labor and costs between the units. Work was divided based on the idea that the central group would be responsible for the platform and business units for developing add-ons and customizing and configuring the software. The development costs were divided according to contracts between the central unit and business units. "*Yeah, it's an agreement per year, upfront principles. We agree for next upcoming budget year and then on monthly basis we get our investment money.*" –Program Manager, central group

Mobilization. The change was managed in a rather top-down fashion, and the benefits were captured, in many cases, in the business units rather than the central unit. "[*The community way of working] is most beneficial for business units where there are a lot of different groups who can contribute or use the contribution. It is a way of improving the components which can be used by others, or we think that others can contribute."* –Business Architect

The benefits of a more distributed setting would result in know-how being closer to the business units and thus in the business units having more say in how the software was developed.

At the time of the interviews, the company was building a new system to allocate the development and maintenance costs between units in a more "just" way. The previous method of "component tax" was not considered adequate, and the company was building an internal software marketplace to share the assets in a better way. This might also serve organizational needs to determine the cost of the different software assets. Should they be, for example, outsourced later?

It is not difficult to track down how the term FLOSS changed: Ideals of meritocratic contribution and voluntary task assignment morphed into contracts among business units, budget negotiations, and steering group meetings. "*We don't really have a community as such. Of course we have a community in a sense that a lot of people working…but it is not an active [FLOSS] community in a way that you probably mean it.*" –Manager

Case 2: Implementing More Open Project Hosting and FLOSS Practices Internally

The Case 2 company is a mobile-communications company employing over 50,000 people globally. Its customer base consists mostly of operators. The company focuses on the production and maintenance of telecommunication network equipment. An internal source-code portal was created in 2003. This portal enables the use of FLOSS practices and tools within the organization. It includes version control tools (Subversion, CVS), issue tracker, mailing lists (Mailman), forums, and file management.

Interpretation. Originally, the internal portal was a response to the growing need to address issues related to hosting and the reuse support of software assets. "*But it's very difficult to find source code that you really can reuse. And if you take it somewhere, will it really work? If you do it by yourself, you know how it works. […] there must be really clear interfaces that you can trus*t." - Senior R&D Engineer

The portal was intended to (1) build on the familiarity the developers already had with FLOSS tools and practices, (2) streamline and standardize a transparent set of tools and software development practices, and (3) enhance collaboration across units made possible by the wider visibility of the source code. *"Everything that we are putting into the portal is ours. It can be used inside the company freely, but nobody else can use it."* –Service Manager

The projects attract global participation. Business units value the version control, quick set-up, and the possibility to support agile projects. There are several ways to use the portal. Common examples include (1) a transparent version control tool, (2) an internal reuse marketplace for software asset showcasing, and (3) a set of tools supporting collaborative software-development practices.

Legitimation. When examining how the events had unfolded after the fact, it is clear that the portal was meant to test the benefits of open source inside one large organization. "*We are using these version control tools from [FLOSS world] and also other tools […] but when it comes to the software that we are selling then it is Inner Source. The most important thing is to share the information what we have*

implemented inside our company. It is then easy to see if someone has already implemented a feature, and it would also be easy to take these features into other projects." –Service Manager

The interpretation of the tool was based on it enabling access to certain good sides of FLOSS, namely reuse and collaboration. It was legitimated as a proven lightweight portal with certain benefits to those projects and developers that were willing to use it. *"[…] but the end customer is not really interested in how we deliver the product, so they are not interested if we are using some Open Source, our portal or whatever, they just want to have a good quality product."* - Senior R&D Engineer

Also, price and familiarity were mentioned as reasons to favor a FLOSS-inspired solution. *"FLOSS is, of course, cheaper for us to use. Engineers who are coming in are quite familiar with FLOSS tools since they have used them in universities or in FLOSS projects."* - Senior R&D Engineer

Mobilization. Mobilization was also based on voluntarism, but an internal service unit was created to give the portal institutional credibility and to promote it internally. This unit has a steering group and a budget, and it serves a key user network consisting of portal users all around the case company. The business units fund the service unit based on the use of the portal. At the time of the interviews, service level agreements (SLAs) were negotiated between the business units and the portal service.

The organization was a lightweight environment compared to some of the more rigid development tools in use at the company. Business units have the autonomy to select whether to use the portal or other internal or external services for their projects. This resulted in a service being driven from the bottom up from business units and by individual developers. *"We need to have some named people behind this service who have the money to pay for the service internally. This has been difficult for a global service because the interest is coming bottom-up. There is no top-management layer making decisions to use to the tool.*" –Global Concept Owner

The technical infrastructure that supports the development process is an almost exact copy of the development process used in FLOSS communities on the Internet. Instead of FLOSS, the respondents tended to talk about inner source, so there is a development similar to the first case. The term open-source software morphed from a published source code and community-driven open collaboration into internal collaboration based on development and maintenance costs divided between business units, with some added visibility of the source code. *"I think that our community is only a set of projects*." –Global Concept Owner

At the same time, a new software-production organization reflecting certain FLOSS practices was formed inside the organization. But, as one respondent noted, unlike in Raymond's *Cathedral and Bazaar*, "*Ultimately, we are paid by work hours.*" – Senior R&D Engineer

Case 3: Controlling the Development of a FLOSS Tool

The Case 3 company (same company as in Case 1 but different business unit) employs approximately 31,000 people. The customer base consists of medical professionals and patients. The company was developing hospital equipment interoperability validation tools that were based on a standard. A rather autonomous business unit of about 2.5 people was dedicated to the development of the testing tool.

Interpretation. The toolkit was originally launched and developed in 2000 and was a proprietary software package developed by two different companies as proprietary software. However, the proprietary license failed due to fears of partiality. The term FLOSS offered interpretation for the publication of the software and a starting point for a collaboration activity, which aimed to gather outside contribution to the development of the testing tool. "*I think the most important benefit [the community brings] is that users use the application and provide feedback about bugs they find."* - Project Leader

Legitimation. FLOSS provided legitimation by showing how the logic of collaboration works in some cases outside a company. The company was moving in a similar direction, although it had encountered problems in attracting outside contribution. "*Currently the most decision making is still done on…on a non-open source way of working."* –Project Leader

The user community could be divided into two groups: those who were the company's customers or using the testing tool and those who were not. The feature requests of the first group seemed to have a priority. *"[Development] is driven by the requests of new functionality from our user community. If there is*

someone who really is expecting, or is a need for certain solution, then at that moment we built a new alpha release." –Project Leader

Currently, most of the communication with users happened via the project's website, which was also maintained by the same business unit. "*The website is where the community comes together. They can ask questions on the forum, they can download the software. If they find a problem, they submit problem reports.*" –Software Engineer

Mobilization. The organization was mobilized with the promise that if certain steps were taken, the project would gain outside contribution and thus better quality and lowered development costs. These steps included organizational issues like moving and creating development discussion channels to the voluntary developers and creating incentives for outsiders to participate. It should be noted that this process was still ongoing and currently almost all development work took place in the original companies that had established the network. *"We don't have anyone else outside these companies that is constantly working on [our product]."* –Software Engineer

The business unit had a steering group, which decided on the strategic issues offline. *"We have within the project a steering committee in which the main contributors come together. In the steering committee, it is decided what we put related to functionalities in the next beta release."* -Project Leader

The funding for the unit came from the different business units that were benefiting in sales from the offered test suite.

The publication of the source code led to several changes in how the software production was organized, from the role of a provider to a joint developer participating in the community providing the software. The company aimed to ultimately shift most of the development and maintenance to an active user community. The developer community is mainly driven by the original initiators and contributors who serve as gatekeepers.

The term FLOSS was used to describe software production, which changed from FLOSS as a voluntary, open, joint collaboration to the direction of a project with a clear management rationale developed mainly by company employees. The source code is published, though it is mainly provided, hosted, and controlled by company employees.

Case 4: Starting a Company Based on FLOSS Development and Consulting

The Case 4 company is a small, Finnish FLOSS firm that specializes in developing knowledge-management and collaborative-learning software, related training, and consultancy. Founded in 1998, the company has its headquarters in Helsinki. The company employed three people at the time of the interviews but has grown since. The company's revenue model is based largely on service contracts with public organizations. These contracts cover software updates, new feature development, support, and training.

Interpretation. The main software product of the company is licensed under an Open Source license; thus, the founders did not hesitate to proclaim the case as a 100% FLOSS company. What about the motivation? The founders thought *"starting a company would be a good way to earn some money as entrepreneurs. We started early by providing training (for example Microsoft Office) to schools.*" –CEO

Originally, the company was engaged in hobbyist web design and selling educational services to schools.

The original choice to base the company's revenue model on FLOSS and focus on public-sector schools happened almost without elaborate planning. *"We had been using Linux internally and the server solutions we offered were based on Linux. Our LAN services included some of our own technology. We got interested in learning environments and offering more services to schools and thus got into that business."* –Marketing Manager

It should be noted that although the source code is published, most of the expertise remains in the case company, whose developers also make most of the development decisions.

Legitimation. There are several different justifications for the choice of the licensing scheme and thus the revenue model. *"We decided to make our own [FLOSS] product. Originally Linux was only a tool for us as starting software entrepreneurs. But later, when we got more interested about the entire*

philosophy, we enrolled with the free-software movement. We were reading Eric Raymond's Cathedral and Bazaar *in 1999 [about FLOSS] and started thinking how we could build our business model on top of that.*" –CEO

"We considered if we could release it as FLOSS. But even in the beginning we saw that we could not compete with the big players who were also moving into eLearning." –CEO

The big competitors, increasing development cost, and gaining credibility in the marketplace all pushed toward drawing on a larger pool of external resources. The respondents agreed that they probably lost some licensing fees because of the license decision, but *"license fees form a very small part of the actual price of the product or the total cost of ownership."* –Marketing Manager

Mobilization. In the case of a small start-up company, it is easier to make dramatic changes in the business model, especially if the software product is new. There are less established user bases and hierarchies than in more established companies. *"A more ideological reason is that we have gotten into this Open Source spirit. As a software developer, you cannot close out ideological views, although business comes first. We would not have gotten involved and continued building our business on FLOSS if we would not have believed in it!"* –CEO

The software company's customers were another group whose reasons and benefits related to FLOSS were discussed. "*Not all of our customers share the FLOSS spirit. Some just think: Hahaa, we are getting this for free! But well, that is not the point actually. Someone still of course has to make the effort to make the free software. The point is not to make everything free.*" –Marketing Manager

The case shows an example of how Raymond's (1999) ideas of FLOSS travel over to a company's software production. But even in the case of a small start-up company, it is possible to identify changes in the meaning of the term. There is a clear idea that the company controls the development, draws on external resources, and ultimately manages the expertise to develop the software. When comparing how this small company views FLOSS to some of the other cases, it is clear that they have a common origin, but their application is very different.

Discussion

Retrospectively, it is easy to identify how the above organizational changes have taken place and how the local meaning of open source shifted from open, community-driven development and widely shared source code into something else. All the cases were, however, directly inspired by open development (tools and practices) and often, similar or the same tools and practices were also used in the organizations. However, the meaning of the term FLOSS changed in the implementation.

There is much variety in what FLOSS can mean to an organization, and the local meanings for FLOSS are empirically very different. Thus, findings urge caution when providing normative advice regarding FLOSS in the organizational setting, especially if it is given in general terms. FLOSS is still far too often seen as a heterogeneous "methodology" or something that could be implemented to software production by using "the correct business model." However, this is an oversimplification of what happens when organizations engage in the process of adopting FLOSS.

Findings show that there are several ways to implement practices termed "open source software" to organizations. The openness of the code base, communications, and access to the tools may be limited to one organization or selected partners. While many FLOSS proponents would contest these practices or their openness, the organizations seem to be benefiting from increased collaboration.

This paper has several important limitations. The analysis is limited to the process that changes the term FLOSS and describes the organizational transformation quite briefly. This is, in part, because showing in detail how the organizational change unfolds is, by necessity, quite difficult, especially if we are following how one buzz-wordish ICT innovation carried into use in huge multinational environments is characterized by hierarchical constraints, paid development work, and constant flux.

The main contribution of the paper to the research on FLOSS is to take the call (Fitzgerald, 2006) to research FLOSS in organizations seriously and reorient and provide accounts of how organizations are leveraging FLOSS. This requires complementing individual developer and macro-level explanations with research on FLOSS in organizations and hierarchical settings. More research focusing on the empirical

changes in the structure in organizational software production might prove to be a fruitful area for further study.

To conclude, FLOSS offers several interesting avenues for further investigation. A better understanding of the rhetorical parts of the adoption process offers insight on how to better manage the process and, indeed, understand the changes FLOSS can bring about in society. FLOSS research has inherently interesting connotations related information commons, but these speculations need to be based on rigorous analysis of organizations – both commercial and noncommercial. Based on our cases, we hesitate to call these FLOSS implementations commons or even commons based.

The next steps in the development of this research project include a new theory of commons that better accounts for these kinds of collaborations and artifacts. Specifically, the tension between hierarchical organizations and provision of commons warrants more research. Novel tools and frameworks are also needed to show how to develop software in the environment characterized by these kinds of hybrid commons.

References

Bauwens, M. 2005. *The Political Economy of Peer Production*, http://www.ctheory.net/articles.aspx?id=499

Benkler, Y. 2007. *The Wealth of Networks*. Yale, CT: Yale University Press.

Castells, M. 2005. *The Rise of the Network Society* (2nd ed.), Malden,MA: Blackwell.

Dahlander, L., and Magnusson, M. (2008). "How do Firms Make Use of Open Source Communities?," *Long Range Planning* (41: 6), pp. 629–649.

Fitzgerald, B. 2006. "The Transformation of Open Source Software," *MIS Quarterly*, (30:3), pp. 587-598.

Gacek, C., and Arief, B. 2004. "The Many Meanings of Open Source," *IEEE Software*, (21:1), pp. 34-40.

Hecker, F. 1999. "Setting Up Shop: The Business of Open-Source Software," *IEEE Software*, (16:1), pp. 45–51.

Hess, C., and Ostrom, E. (2007). Introduction: an Overview of the Knowledge Commons", In. *Understanding Knowledge as a Commons: From Theory to Practice* Eds. Hess, C. and Ostrom, E. Cambridge, MA: MIT Press.

Klein, H. K., and Myers, M. 1999. "A Set of Principles of conducting and Evaluating Interpretative Field Studies in Information Systems," *MIS Quarterly*, (23:1), pp. 67-94.

Osterwalder A., Pigneur, Y., and Tucci C. 2005. "Clarifying business models: Origins, present, and future of the concept," *Communications of the Association for Information Systems,* 16, pp.1-25

Ostrom, E. and Schlager, E. (1996). The Formation of Property Rights Eds. Hanna, S., Folke, C., and Mäler, KG. *Rights to Nature*, Washington D.C., Island Press, pp. 127-156.

Raymond, E. 1999. *The Cathedral & The Bazar - Musings On Linux And Open Source By An Accidental Revolutionary*, Sebastopol, CA: O'Reilly Associates.

Rolandsson, B., Bergquist, M., and Ljunberg, J. (2011). "Open source in the firm: Opening up professional practices of software development," Research Policy (40: 4), pp. 576–587.

Scacchi, W. 2007. "Free/Open Source Software Development: Recent Research Results and Methods," In Zelkowitz, M. V. *Advances in Computers*, Academic Press. 69, pp. 243-269.

Shaikt, M., and Cornford, T. 2010. "'Letting go of Control' to Embrace Open Source: Implications for Company and Community," in the proceedings of HICSS2010, Kauai, Hawaii.

Sharma, S., Sugumaran, V. and Rajagopalan, B. 2002. "A framework for creating hybrid-open source software communities," *Information Systems Journal*, (12:1), pp. 7-25.

Swanson, B., and Ramiller, N. 1997. "The Organizing Vision in Information Systems Innovation," *Organization Science*, (8:5), pp. 458-474.

Välimaki, M. 2005. *The Rise of Open Source Licensing*. Doctoral Dissertation of Helsinki University of Technology, Helsinki: Turre Publishing.

Von Krogh, 2002. "The Communal resource in information systems*," Journal of Strategic Information Systems*, (11), pp- 86-107.

Walsham, G. 1995. "The emergence of interpretivism in IS research," *Information Systems Research*, 6(4), pp. 376–394.

Žižek, S. 2009. *First as Tragedy Then as Farce*, London:Verso.